PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 13/00	A1	(11) International Publication Number: WO 99/22301
		(43) International Publication Date: 6 May 1999 (06.05.99)

(21) International Application Number: PCT/US98/22598

(22) International Filing Date: 23 October 1998 (23.10.98)

(30) Priority Data:
08/957,652 24 October 1997 (24.10.97) US

(71) Applicant: CELLPORT LABS, INC. [US/US]; Suite 300E, 4888 Pearl East Circle, Boulder, CO 80301 (US).

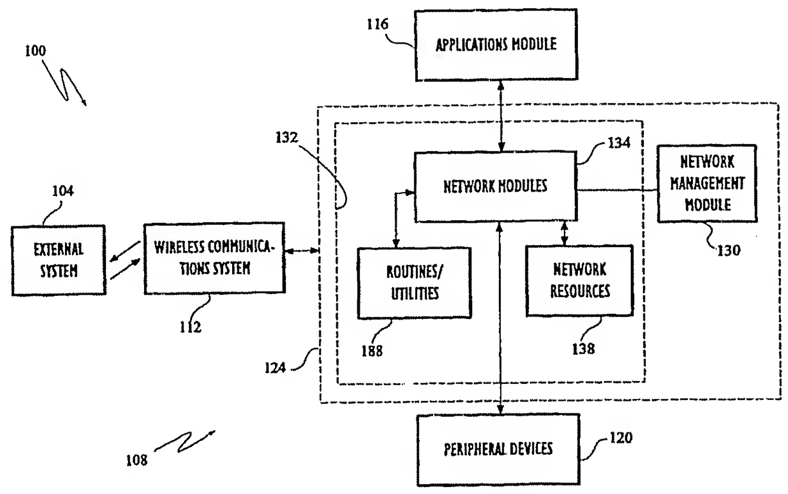
(72) Inventors: BENTLEY, James; 11213 Vilas Street, Parker, CO 80134 (US). KLINGENSTEIN, Kenneth, J.; 8454 Boulder Hills Drive, Longmont, CO 80503 (US). BRAITBERG, Michael, F.; 440 Broken Fence Road, Boulder, CO 80302 (US). SPAUR, Charles, W.; 3911 South Elkhart Street, Aurora, CO 80014 (US).

(74) Agents: ZINGER, David, F. et al.; Sheridan Ross P.C., Suite 3500, 1700 Lincoln Street, Denver, CO 80203-4501 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: COMMUNICATIONS SYSTEM WITH MODULAR DEVICES



(57) Abstract

Apparatus and method for communication between an external system (104) and a communications apparatus (112), preferably embedded in a vehicle and including a modular base system (124) comprising a number of network devices (132) and a network management module (130). The network devices (132) include network modules (134) and network resources (138). Each network module (134) includes a number of subsystems that provide communications-related functions (188) and one or more of them is part of different network layers. The network modules (134) are able to communicate with each other using a common interface (142) under the management of the network management module (130). The network resources (138) include first in first out buffers, circular buffers and no copy buffers. The base modular system (124) communicates with an applications module (116) and physical devices (120) located in the vehicle. After network module registration with the network management module (130), communications can be made utilizing the modular base system (124) between the applications module (116) and the physical devices (120).

COMMUNICATIONS SYSTEM WITH MODULAR DEVICES

FIELD OF THE INVENTION

The present invention relates to system and method for communicating between an application program and a device through multiple communication layers and, in particular, for communicating using one or more selected network modules at the different communication layers of the modular system.

BACKGROUND OF THE INVENTION

Application programs are commonly involved with the transfers of information including data from/to a variety of devices including physical devices. Such information is typically used by the application program in conjunction with its processing-related functions. The transfers of information can involve a number of different network layers or levels. These network levels might include device drivers, device protocol drivers, routing protocols, transfer control protocols, sockets and other network links. It is commonplace to establish a communication path or channel using a particular defined set of network levels when transferring information between an application program and a device. That is, for a particular application and device, a single communication path or channel involving defined network levels is provided to handle desired information transfers. By way of example, for an Internet communication, network levels comprising an Ethernet device driver, an Ethernet protocol driver, TCP/IP (transmission control protocol/Internet protocol) and a socket interface can be operatively linked together.

In a pending patent application assigned to the same assignee as the present invention, a portable mobile communications system is described that includes a communications apparatus that can be embedded in a vehicle for controlling information transfers between physical programs. The base system provides the communications capability between the peripheral devices and the applications. Such communications include transfers of information including data between one or more of the peripheral devices and one or more of the applications. The base system is also involved with controlling and managing communications between the external system and the peripheral devices. Such communications can include the sending of information using results from running or executing an applications program, by means of data directly transmitted between the external system and one or more of the peripheral devices, and/or the downloading of applications programs. In one embodiment, the external system includes one or more remote stations that are located at a distance from the base system and one or more of the remote stations may utilize different communication protocols when accessing the base system. In such an embodiment, communications may occur wirelessly over an airlink to the base system, particularly where the base system is located at a great distance from the particular remote station.

The modular base system includes a network management module and a number of network devices that are controlled and/or managed by the network management module. The network devices include network modules in which at least some of them can be properly linked together to establish a communication path or channel between an application and one or more peripheral devices, such as peripheral devices found in or associated with a vehicle. These network modules include a number of well-known communication devices, such as sockets, TCP/IP, device protocol modules including Ethernet and device drivers that support the particular peripheral or physical device. Such network modules also include an address resolution protocol (ARP) module (typically combined with a reverse address registration effectively renders the base system transparent to the code employed by the user when accessing the base system.

The common interface allows each network module to be connected to other network devices which implement and/or expect this interface. Each module implements a common interface of open, read, write, set options, get options and close functions or routines. Each such routine of each network module is provided to the network management module in connection with its management functions.

The network devices of the base system also include a number of network resources that are created by one or more of the network modules to assist them in their communication tasks. These network resources include one or more FIFO (first in first out) buffers. A FIFO buffer is useful in establishing queues for task or network module management. The TCP (transmission control protocol) module may create a FIFO buffer in order to provide a resource that is useful in its responding to connection requests from one or more network modules in another layer, such as the socket interface module. Other network resources include circular buffers (CBs) that have been determined to be flexible and effective in the organization and transfer of data within and between network modules. The IP (Internet protocol) module might utilize such a circular buffer to store route information related to one or more particular network devices that it communicates with. Still other network resources include one or more NCBs (no copy buffers) that are provided to reduce resource utilization and the need for copying data between network modules, which enhances throughput and responsiveness. In particular, the NCB can be passed bi-directionally between and among different network modules of the different layers.

(system network management protocol) functions. The security module is used in authenticating and/or approving network connections as part of a firewall protection scheme. The accounting module enables users to obtain information related to network or other usage costs when utilizing the modular base system 124 and can use such information to reconcile it with their own usage records.

With regard to the operation of the modular base system, particularly involving the interactivity among and between network devices and the network management module, such operations or involvement can be categorized as involving registration of network modules, including resolutions of conflicts among network modules during the registration process and obtaining maximum transfer unit (MTU) information from network modules, together with creation of network resources and using them for information transfers. The registration of network modules with the network management module comprises the supplying of information by the registering network module to the network management module. This information includes: the protocol identification (ID) field, which refers to the assigned protocol numbers as provided in RFC (Request for Comments) 1700 ; the maximum transfer unit (MTU) field that represents the largest number of data, such as bytes, that the network module can send or receive; the network module's header information; and one or more of a number of routines for communication involving network modules, such as the open, close, read, write, set options and get options routines. The network management module returns a unique identification to the network module that the network module will use in its communications with the network management module. Additionally, the network management module will maintain a list of the particular network module's registration information that was supplied. The network management module will also keep a manipulate options and close the opened interface with the second network module. Registered error handling routines can also be called by network modules when they experience errors during operation. A predetermined call function is utilized for invoking a particular or desired error handling routine. In response, the error handling routine takes appropriate steps in accordance with its algorithms or predetermined steps. Similarly, a network module, such as the IP module, may invoke a particular routing routine, in conjunction with or separate from, the execution of the error handling routine. In one embodiment, as part of a routing routine, a channel scheduling module is employed related to controlling or buffering information that is to be transferred at an appropriate or available time. Fundamental steps are also taken when there are communications involving network resources. In the context of the TCP (transmission control protocol) module, it is invoked, for example, when a first connection request is received, with this first connection request being queued in the FIFO buffer and indication is given thereof. When a second connection request, for example, is received from another network module, it is also queued in the FIFO buffer. Subsequent reads of the FIFO buffer result in the first connection request being read or accessed before the connection request that arrived second. A CB can be invoked for accessing route information stored in a predetermined location in the CB. The NCB is defined to include a total number of bytes. When conducting the allocating function, data to be transferred is copied into the NCB. The data occupies less than the total number of bytes. The NCB with the copied data bytes can then be passed from a first network module to a second network module. When the NCB is passed, a pointer is adjusted from a first pointer position to a second pointer position. The second pointer position relates to the header information particularly when taken together with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of the present invention illustrating major communication units;

Fig. 2 diagrammatically illustrates the network management module, network modules and network resources of the modular base system, as well as utility routines and operating system construct;

Figs. 3A-3B illustrate details of the network management module and representative network modules related to information lists, that are generated and stored using the network management module, having parameters useful in communication transfers and also illustrating representative network modules and their common interface;

Fig. 4 diagrammatically illustrates the interaction between and among network devices and the network management module in connection with information transfers among network layers in the modular base system; and

Figs. 5-9 constitute flow diagrams illustrating major steps that are conducted as part of information transfers among network devices of the network layers in conjunction with transferring information relative to one or more applications and one or more peripheral devices.

DETAILED DESCRIPTION

Referring to Fig. 1, a communications system 100 is illustrated, which includes an external system 104 and a communications apparatus 108, such as an embedded system, that communicate with each other using, preferably, a wireless communications system 112. In one embodiment, the embedded system 108 is embedded in a vehicle or other mobile unit to permit bi-directional communications from/to the vehicle. A disclosure directed to such an embedded external system 104 when communication transfers are occurring in that direction.

The embedded system 108 includes an applications module 116 having user programs with communication requirements relative to one or more sites different from the vehicle. That is, the external system 104 might require certain data or other information from the vehicle and one or more user programs function to provide such information. In that regard as well, the embedded system 108 includes one or more devices 120 located in or associated with the vehicle including physical devices (all such devices referred to as "peripheral devices"). By way of examples, such peripheral devices 120 can include one or more of the following: GPS (global positioning system), dead reckoning sensors, facsimile machine, PDA (personal digital assistant), laptop computer, printer, display unit, modem, CD-ROM unit, storage device(s), medical condition sensor(s), vehicle condition sensors, vehicle cargo sensor(s), air bag sensor and general alarm sensors related to unlocking the vehicle or unwanted intrusion into the vehicle. In one embodiment, one or more such peripheral devices 120 may communicate with and be under the common control of a vehicle controller area network (CAN) as will be discussed later herein.

The embedded system 108 also includes a modular base system 124 for use in providing information transfers between the applications module 116 and the peripheral devices 120. The modular base system 124 is modular in design so that the modular base system 124 can readily adapt to the addition of later designed communications-related modules that provide different and/or additional beneficial functions. Furthermore, such modularity facilitates testing of units separately in the modular base system 124. The modular design also enables the user or module 142, especially where a particular application is not suitable for this socket interface.

With reference to Figs. 2 and 3A-3B, more information and greater details are next provided related to the modular base system 124. As seen in Fig. 2, there are a number of network modules 134. All or a substantial number of them can be defined as being part of a particular one of a number of network layers. The network layers are defined similar to International Standards Organization (ISO) network layering, such as in accordance with the following:

NFS/HTTP/FTP/TELNET/RLOGIN/USER	Highest Network Layer
TCP/UDP Modules	↑
IP/ARP/RARP/ICMP/IGP/EGP/BGP Modules	↕
Device Protocol (Ethernet SLIP, PPP, 802) Modules	↓ Lowest Network Layer
Device Driver Modules	

With regard to such network layers, it is anticipated that data/information cmpression is likely to occur at a number of the network layers, as well as in the applications. Such data compression will include the option to disable compression on lower network layers if such would impact higher-level applications. As an example, in the context of video data, under MPEG II, certain frames may not be suitable for compression, while streaming JPEG will be compressible again at a lower network layer.

Each of such network modules 134 has a common interface associated with it that enables each of them to communicate in a common manner with other network modules 134, on the same or different network layers. With regard generally to such network modules 134, each device driver module 146 supports the physical implementation of the one or more peripheral devices 120 that it is operatively connected to. A network layer above the network layer particular communications channel is available for use; a communications channel scheduling module 188, which commonly works with the channel controller/monitor 186, and functions to retain or store information until a particular communications channel is available. Other support modules 182 include: a data encryption module 190 for use in encrypting data before transmission, a data compression module 192 for preparing data in a reduced number of bytes for transfer and a modem control module 194 that is available for assisting in modem transfers; a network time module 196 for providing necessary time-related information for routing functions and system network management protocol; a security module 198 for insuring only proper and authorized access to the modular base system 124; and an accounting module 202 involved with the tracking of usage of the modular base system 124, particularly in the context of making suitable information available that is useful to the user in connection with monitoring or maintaining network usage costs.

Other network devices 132 having special functions for performing a particular task when certain or predetermined circumstances occur are defined generally as routines/ utilities 202 and include: error handling routine(s) 206 , information gathering routine(s)208, routing routine(s) 212, callback routine(s) 204 and checksum routine(s) 210. An error handling routine 206 is invoked when an error occurs, such as during the operation of a network module 134 and which module wishes to report the error and have the error handled by this routine 206. The information gathering routine 208 provides statistics-related data concerning error reporting and/or other functions of network modules 134. A routing routine 212 is beneficial in ascertaining a suitable or optimum communications path or channel for a particular information transfer. In one embodiment, such a routing routine 200 includes the communications. The socket interface module 142 of the present invention is based on the known Berkeley style socket routines. In connection with communications with network modules 134, the sockets of the socket interface module 142 rely on the common interface routines of the network modules 134, particularly to gain port numbers used in lower network layer protocol communications. The sockets of the socket interface module 142 may directly manipulate port structures returned to it or such sockets may make TCP module 174 and UDP module 178 calls using the common interface routines of these network modules 134.

The TCP module 174 is implemented or structured based on the standard RFC 793 including the specifications detailed therein. The TCP module 174 also has the common interface including access to routines useful with one or more sockets of the socket interface 142 for retrieving an instance of the TCP module 174. In conjunction with its operation, the TCP module 174 is invoked at initialization of the modular base system 124. The TCP module 174 is invoked after the IP module 154 is loaded and before implication of the services associated with the socket interface module 142. As applications use their instances of transmission control protocol, their state and current data values may change. To accommodate the multiple threads of execution, the TCP module 174 must be re-entrant. The TCP module 174 will use data blocks to hold application state/port information and associate the appropriate data block to the instance of the TCP module 174.

Similar to the TCP module 174, the UDP module 178 is based on previously established standards, namely, in RFC 768 and the specifications detailed therein. In addition to having the common interface, the UDP module 178, as will be explained further herein, must provide and register its own open routine to allow sockets of the socket interface each interface of the network module 134 that is registered. On the other hand, there are instances where an application wants to control information or data transfers, in the context of the routing that such information should take, and/or the physical media to be employed, for example. In that regard, security concerns may require a particular route for such an application and the modular base system 124 includes this capability for permitting such controls to be established by the user.

When invoking the IP module 154, the length of the invocation might be so long that a configuration file may be necessary. The IP module 154 constructs a network device table based on the network devices 132 passed to it. Data associated with such network devices includes: the network device 132 name, the IP address, the network mask and the device type given with the name. Additionally, the data could include the network device's maximum transfer unit (MTU) related to the amount of data that the network device can handle and header length that relates to the amount of information comprising the header of the particular network device. The IP module 154 is able to call other protocol modules defined as being part of its network layer including the ARP/RARP, IGMP, BGP, EGP and IGP protocol functions when they are registered with the network management module 130. The IP module 154 reads and writes from a wrapper routine and such a routine will act as an insulator from the actual underlying physical read/write constraints. The IP module 154 implements blocking operations, for protocol to Internet protocol communications, using event constructs including events related to pending information receptions and pending information transmissions.

The ARP/RARP (address resolution protocol/reverse address resolution protocol) module 158 provides the ability to decouple Internet protocol from the underlying a PPP module 262,a DHCP (dynamic host control protocol module 264, an ATM (asynchronous transfer mode) module 266, a 802 module 268, and a CARP (controller address resolution protocol) module 270. Ethernet and Arcnet protocol modules 250, 254 have well-established protocols, and their implementations as network modules correspond to such protocols. Like other network modules 134, each has a common interface for proper communication with other of the network modules 134. The SLIP module 258 is based on standards set forth in RFC 1055 and protocol specifications detailed therein. The SLIP module 258 has the ability to provide the CSLIP functionality. The SLIP module 258 is able to operate in blocking and non-blocking modes, as well as operate in compressed or non-compressed modes. One form of invocation of the SLIP module 258 involves the use of command line arguments that may be executed by currently available CPMW code. Such a command line argument would include the name under which the SLIP module 258 registers itself. A second command line argument indicates the name of the network device 132 over which the SLIP module 258 will read and write. Another optional command line argument specifies whether or not to use compression for the registered network device 132. The PPP module 262 is based on the standard RFC 1661 and the protocol specifications detailed therein. The PPP module 262 is responsive to dynamic Internet protocol assignment and dial-on-demand PPP operations. As with other network modules 134, the PPP module 266 registers itself with the network management module 130. The PPP module accepts a name over which to read and write and determines if the link is to be compressed. When dynamic IP addresses are provided, the PPP module 262 is given the IP address which it will assign or base assignments on. The DHCP (dynamic host control protocol) module 264 may be accessed and is useful in dynamically assigning IP addresses for be used for communication purposes outside of the modular base system 124.

With reference particularly to Figs. 3A-3B, further descriptions are provided regarding the registration of network modules 134 with the network management module 130, as well as establishing communication paths that involve the network modules 134 and network resources 138. Each network module 134 is required to register with the network management module 130. The network management module 130 is able to monitor network resources 138 required at each network level, for each protocol of an associated network module 134 and dynamically resolve network module 134 interface procedure calls. In one embodiment for registering with the network management module 130, a call identified as net_register is utilized and defined as:

net_register(name, network layer, registration structure).

Each network module 134 sends the network management module 130 a name to identify itself, particularly related to its protocol. The network management module 130 also requires the network module 134 to specify its network layer or level on which the particular network module 134 expects to operate. The network management module 130 additionally is useful in resolving inconsistencies or conflicts that might arise between or among network modules during the registration process.

Each network module 134 also provides a set of values upon registration with the network management module 130, namely:

protocolID - reflects the protocol's assigned number as set forth in the Internet publication "Assigned Numbers," RFC 1700;

MTU (maximum transfer unit) - represents the largest number of bytes (octets) that the particular network module 134 can send or receive;

modular base system 124 to set network module specific options as determined by the option code, option value (typically set or reset) and option value size (for determining data type and data space expected). Get_option allows a user to verify currently set module specific options, with option settings being determined by the option code provided, returned in the option value space provided and is of the option value space size that is also provided. Close is used to remove the user initiating the close from the active users that the network module 134 is accommodating so that any space held by associated data structures are made available and so that a user is properly removed from conducting network module activities. Return codes indicate success or failure of each operation. ARG (argument) constitutes a value or other information determined by each network module 134 to help each such network module 134 identify the user and/or necessary values to carry out each operation.

As depicted in Fig. 3, such information or values are stored in a linked list by the network management module 130, together with additional items. In one embodiment, such registration information can be defined by the following linked list structure:

```
guard;
name;
network layer;
registration structure;
node_next; and
node_prev.
```

The information or values associated with name, network layer and registration structure data have already been described. The guard field is set to a unique value for validity checking. The node_next and node_prev fields are used to store pointers that relate to the elements call, the calling network module 134 is able to make interface calls for the purposes of transferring data, manipulating options and closing the opened interface. The calling network module 134 uses an identifier returned from the net_open call when making any manipulation calls to the called network module 134. In one embodiment, the messages or calls that are invoked related to communication functions between two network modules 134 can be defined as:

net_close (ID);
net_write (ID, data, DataLan);
net_read (ID, DATAP, DATAPLAN);
net_set_option (ID, option, void optval, optvalsize); and
net_get_option (ID, option, void optval, optvalsize).

Net_close calls the registered CloseRoutine associated with the ID field. The net_write calls the registered WriteRoutine associated with the ID field. The net_read calls the registered ReadRoutine associated with the ID field. The net_set_option calls the registered SetRoutine associated with the ID field. The net_get_option calls the registered GetRoutine associated with the ID field.

With continued reference to Fig. 3A, the FIFO buffer 216, the CB 220 and the NCB 224 of the network resources 138 will be further described. Referring first to the FIFO buffer 216, it is useful in communications internal to as well as between network layers. The FIFO buffer 216 is accessible using the previously described OpenRoutine, CloseRoutine, SetRoutine, GetRoutine, WriteRoutine and ReadRoutine. Regarding the creation of a FIFO buffer 216, a network module 134 is typically involved and uses a predetermined function having a number of parameters, such as:

net_crt_fifo (n_elts, name)

```
size;
struct FIFOelt next;
struct FIFOelt prev;
data.
```

The foregoing structure will be dynamically created to be the size of the buffer passed (as indicated by the buffer size argument given in the call), plus the size of the netFIFOelt structure. The buffer size will be stored in the size field. The buffer data will then be copied into the created netFIFOelt structure beginning at the data field. The structure will then be added to a doubly linked list using the next field to reference the next element in the list and the prev field to reference the previous element in the list. Netfifoelt structures are added to the list in a manner that the first element added is the first element retrieved.

To cancel or otherwise remove the created FIFO buffer 216, the user of the particular FIFO buffer 216 invokes a predetermined call, such as:

```
net_destroy_FIFO (Id FIFO Id)
```

Upon being invoked, this call will free memory allocated to each element stored in the queue of the particular FIFO buffer 216 being destroyed, as well as unregister itself from the network management module 130 and release any memory associated with the buffer control structure.

Regarding the CB 220, it is useful in communications internal to as well as between network layers having different protocols. Like the FIFO buffer 216, the CB 220 provides the OpenRoutine, CloseRoutine, SetRoutine, GetRoutine, WriteRoutine and ReadRoutine. A user of a CB 220, such as a network module 134, creates the CB 220 before it is used. In one embodiment, a function having parameters that is used to create a CB 220 is defined as:

```
net_crt_circ (size, name)
```

options for faster circular buffer data access, e.g., to implement a linear hash buffer of elements.

A previously created CB 220 can be removed by the user making a predetermined call such as through use of a net_destroy_circ routine identified as:

net_destroy_circ (ID circId)

The circId field is the identifier returned to the user from crt_crc or net_open. The functions resulting from the call to destroy_crc releases the memory held by the circ_buf structure associated with the given circId.

It should be appreciated that recovery or other procedures can be invoked when a particular FIFO buffer 216 or CB 220 experiences an overflow or other buffer usage problem. In such a case, conventional and available techniques or procedures are included for access and usage in order to handle any such event or error that occurs.

Referring to the NCB 224, such no copy buffers are intended to reduce the number of copy operations that occur when transferring data between and among network modules 134. The network management module 130 has an established process for registered network modules 134 to allocate a NCB 224 by utilizing a predetermined function or call identified as:

net_aloc (ID, size)

The ID parameter represents the caller's network identification as returned by a call to net_register. The size parameter indicates the number of bytes the caller would like to allocate. Upon invoking the net_aloc function, a pointer is returned to a contiguous block of memory of the indicated size. Additionally, when calling the function net_aloc, the network management module 130 determines the number of bytes that may prepend the NCB 224 by taking into account previously stored information about the network module 134 that is making the call for the NCB 224, with such information including the number of network

To release memory allocated based on invocation of the net_aloc function, a predetermined call to a routine for providing such a release is made by a network module 134, which call can be identified as:

net_free (ID NCB)

Invocation of the free function traverses the allocation linked list to find an entry containing the reference NCB. The given NCB is matched with a stored NCB by looking at the pointer values. The given NCB pointer value must be greater than or equal to the stored NCB pointer value and less than or equal to the stored NCB pointer value plus the stored NCB size (NCB size). If a match is found, the stored NCB is freed as well as the entry containing the matched NCB.

Further detail is next provided related to use of a NCB 224 in connection with passing or transferring data between and among network modules 134 including the prepending of header information. Such information is provided in the form of an example. Assume that the socket interface module 142 requests 500 bytes. Upon such a request, 554 bytes are actually allocated. The network management module 130 obtains the header information for the network modules 134 that will follow the socket interface 142 in connection with the passing of the 500 bytes of data. In that regard, the network management module 130 obtains the maximum header information (HeaderLen) for the TCP module 174 (assume 20 bytes); for the IP module 154 (assume 20 bytes) and for the Ethernet protocol module 250 (assume 14 bytes). The particular socket of the socket interface module 142 receives a reference within the NCB 224 at an offset of 54 bytes so that only the 500 bytes requested may be directly accessed. The socket interface module 142 is used to copy the 500 bytes into the NCB 224 and passes the NCB 224 to the TCP the network management module 130 and called by the network modules 134 to process errors of the network modules 134. More than one error handling routine 206 can be registered. Each such error handling routine 206 might be called in the order in which it was registered or, alternatively, any such error handling routine 206 could be called or managed in accordance with other suitable rules or protocols. In one embodiment, an error handling routine 194 can be registered by a user calling the following function having the identified parameters:

ereg_hndlr (hndlr, ARG)

A function pointer (hndlr) is given that represents the entry point to the particular error handling routine to be called for processing errors. The ARG parameter specifies information to be given as the first parameter to the called function or routine identified as hndlr. The error handling routine 206 also takes ID, error and char parameters in that order. The ID parameter identifies the network module 134 which generated the error. Typically, the error handling routine 206 calls get_option to get the protocol identifier and other information thereby identifying the caller. The error parameter identifies the error being reported. The char parameter constitutes the specific data for the generated error. The error handling routine 206 determines if the error is informational (statistic) or if it indicates a problem in the system. The return to a dreg_hndlr call provides a network identifier for the particular error handling routine 206.

When a user determines that it wishes to discontinue using a particular error handling routine 206, a predetermined function is invoked, such as that identified by the following:

eunreg_hndlr (ID)

This called function removes the identified error handling routine 206 from the list of active error handling system constructs. In particular, Fig. 4 illustrates a number of selected network modules 134 from different network layers that are involved in a particular application required by the applications module 116. Fig. 5 briefly describes in flow diagram fashion the major steps involved in conducting such operations. In particular, the applications module 116 initially seeks to transfer data to at least one peripheral device 120. In accordance with this objective, the necessary or appropriate network modules 134 and network resources 138 must be registered and operatively opened or connected with and among each other before such a transfer can occur. In that regard, at step 304 of Fig. 5, the Ethernet device driver module 280 registers with the network management module 130. As part of its registration, the Ethernet device driver module 280 registers the identity of its network layer, the size of its header information and its MTU, which indicates the maximum amount of data that can be transferred across its interface at one time. As previously described regarding Fig. 3, common interface routines are also registered and lists are stored related to information or values associated with the Ethernet device driver module 280.

At step 304, the Ethernet protocol driver module 250 acquires an interface for the Ethernet device driver module 280 by use of the appropriately described functions or calls including use of an open call to be able to use an OpenRoutine for establishing the interface connection between these two modules. The Ethernet protocol module 250 is also registered with the network management module 130 and provides the same values, data and/or other information that is required for registration, just as was done when the Ethernet device driver module 280 was registered. Additionally, the Ethernet protocol module 250 registers an event1 214 with the Ethernet device driver module 280. This event will be utilized in connection with and the calling of the open function that resulted in obtaining the OpenRoutine for the IP module 154 from the network management module 130. At step 348, the TCP module 174 registers its name with the network management module 130 and, like other network modules 130, specifies its network layer, header size and MTU, together with other required values or information to be included in the appropriately stored lists. A callback3 204 is also registered by the TCP module 174 at step 352 with the IP module 154. This callback3 204 is useful in transferring data from the IP module 154 to the TCP module 174.

At step 356, the socket interface module 142 calls the registered TCP module 174 and its registered OpenRoutine. The socket interface module 142 specifies a passive call, which leads to the creation of a FIFO buffer 216 for another or returned TCP module interface that has a FIFO buffer 216 for use in accepting connections to be handled by the TCP module 174. Registration of TCP module routines (Read, Write, Get, Set, Close) is also made to enable their use by the socket interface module 142. This returned interface is used for reading and writing functions involving the sockets and the TCP module 174. At step 364, an event2 214 is registered with the TCP module 174 for indicating socket interface communications being received by it.

With these network modules 134, network resources 138, callback routines 204 and event constructs 214 registered or otherwise established, an example related to transfer of data bytes is next described. Beginning at step 368 of Fig. 6, the applications module 116 seeks to transfer data bytes and starts this process by writing the data bytes over the socket interface module 142. At step 372 of Fig. 7, in connection with use of a network resource 138 to transfer the data bytes, the socket interface module 142 requests a NCB 224 with its size dependent on header sizes applications module 116. At step 416 of Fig. 8, this data is received initially by the Ethernet device driver module 280. The event1 208 is set, at step 420, by the Ethernet device driver module 214 indicating that it has data available for transfer. Subsequently, at step 424, the Ethernet device driver module 280 sends the data bytes to the Ethernet protocol module 250 at step 424. Comparable to the description related to the transfer of data from the applications module 116, each network module 134 that is involved with the transfer of the data bytes adds its header to the NCB 224 and adjusts the offset pointer for access by the next network module 134 that will receive the passed NCB 224. In accordance with this procedure, at step 428, the Ethernet protocol module 250 adjusts the offset pointer for the NCB 224 based on its header. Further, based on information that it receives, the NCB 224 is passed to the IP module 154 using callback2 204 and bypasses the ARP/RARP module 158. For a different destination, the Ethernet protocol module 250 accesses the callback1 204 of the ARP/RARP module 158, which is provided to it in the packet of data being transferred. Once a resolution of where the data is to be transferred is complete, the ARP/RARP module 158 then sends the buffered data to that location.

When the Ethernet protocol module 250 sends the data to the IP module 154, it adjusts the offset pointer of the NCB 224 based on its header at step 432. With regard to the transfer of the NCB 224, the IP module 154 invokes the callback3 204 routine that was registered previously by the TCP module 174, in accordance with step 436. The callback3 facilitates the transfer of the data between these two network modules. Upon receipt of the transfer data, at step 440, the TCP module 174 adjusts the offset pointer of the NCB 224 based on its header information. At step 444, the TCP module 174 sets the event2 214 that was registered

In that regard, the channel controller/monitor module 186 is used in monitoring and determining when the new communications channel is ready, in accordance with step 484. With the new communications channel being ready for the data transfer, the channel scheduling module 186 permits the data to be transmitted at step 488. Based on the release of the data using the channel scheduling module 188, the IP module 154, at step 492, transmits the data over the new communications channel. Apparatus and methodology for providing such related error handling and routing functions is described in substantially greater detail in a pending and related U.S. Patent Application identified by Serial No. 08/778,897 filed January 3, 1997 and entitled "Communications Channel Selection". This pending application is hereby incorporated by reference, particularly including pages 11-34, as well as Figs. 1-5B, and even more particularly, pages 18-34 including the referenced drawing figures. Such disclosures in this related application have particular applicability to the present application in conjunction with its description of an embodiment for selecting an acceptable communications channel. In addition to the foregoing functions and operations for communication transfers involving the modular base system 124, further network modules can be designed and employed. Included among these might be a network module 134 used to manage priority of communication transfers and provide preemptive control in connection with communication transfers depending upon the presence of one or more predetermined factors.

The foregoing discussion of the invention has been presented for purposes of illustration and description. Further, the description is not to limit the invention to the form disclosed herein. Consequently, variations and modifications commensurate with the above teachings within the skill and knowledge of the relevant art, are within the

What is claimed is:

1. An apparatus for communicating using a plurality of modules, comprising:

a plurality of network devices including a number of network modules and a number of network resources, with each of said network modules having a common interface; and

a network management module communicating with each of said network modules for registering each of said network modules and for assisting said network modules in establishing said network resources that are useful in communication transfers involving said network modules.

2. An apparatus, as claimed in Claim 1, wherein:

at least some of said network modules are defined as being included in different network layers and in which said some network modules in said different network layers communicate with each other using said common interfaces said network modules.

3. An apparatus, as claimed in Claim 1, wherein:

said network modules include a device driver module, a device protocol module, an Internet protocol (IP) module, a transmission control protocol (TCP) module that is separate from said IP module and a socket interface module.

4. An apparatus, as claimed in Claim 3, wherein:

said network modules include an address resolution protocol (ARP) module that is used in decoupling said IP module from one or more network modules that underlie said IP module in a different network layer.

5. An apparatus, as claimed in Claim 3, wherein:

said device protocol modules include one of an Ethernet module, a SLIP module and a PPP module.

6. An apparatus, as claimed in Claim 3, wherein:

said network modules include an ICMP module, an IGP module and an EGP module, each of which is defined to be on the same network layer as said IP module.

related to header information, module interface routines and module specific information in an argument field.

14. An apparatus, as claimed in Claim 1, wherein:

said network management module is used to store a list, by network layer, of header information of each of said registered network modules.

15. A method for communicating using selected ones of a plurality of network modules and a network management module, comprising:

establishing a network management module;

providing a plurality of network modules including first and second network modules;

registering each of said network modules with said network management module; and

communicating information using selected network modules including between said first and second network modules utilizing a common interface associated with each of said first and second network modules.

16. A method, as claimed in Claim 15, wherein:

said establishing step includes making available communication links between each of said network modules and said network management module and using communication messages having predetermined parameters as part of said registering step.

17. A method, as claimed in Claim 15, wherein:

said providing step includes maintaining communication separateness among said plurality of network modules which requires that each of said network modules be able to communicate with others of said network modules using said common interfaces.

18. A method, as claimed in Claim 15, wherein:

said registering step includes supplying information by said first network module to said network management module, with said information including a plurality of the following: a protocol assigned identification, header

25. A method, as claimed in Claim 22, wherein:

said creating step includes transmitting a name for said CB and a number of bytes that said CB is allowed to hold as part of said parameter information.

26. A method, as claimed in Claim 22, wherein:

said creating step includes allocating said NCB by sending an allocate function to said network management module by said first network module and in which said allocate function has a number of parameters including an identification parameter that represents the identification of said first network module and a size parameter that relates to a number of buffer locations requested for allocation by said first network module.

27. A method, as claimed in Claim 15, wherein:

said communicating step includes said first network module opening an interface to said second network module and then using at least one of the following routines in connection with said communicating step: write, read, set options and get options.

28. A method, as claimed in Claim 27, wherein:

said communicating step includes closing said interface after said interface has been opened with said second network module by said first network module.

29. A method, as claimed in Claim 15, wherein:

said communicating step includes calling an error handling routine by said first network module using said network management module and in which said calling step includes identifying said first network module as generating an error and including a parameter that identifies the error being reported.

30. A method, as claimed in Claim 15, wherein:

said communicating step includes making a call to a callback routine for use in receiving data and in which said making step includes calling a set option routine of said second network module by said first network module.

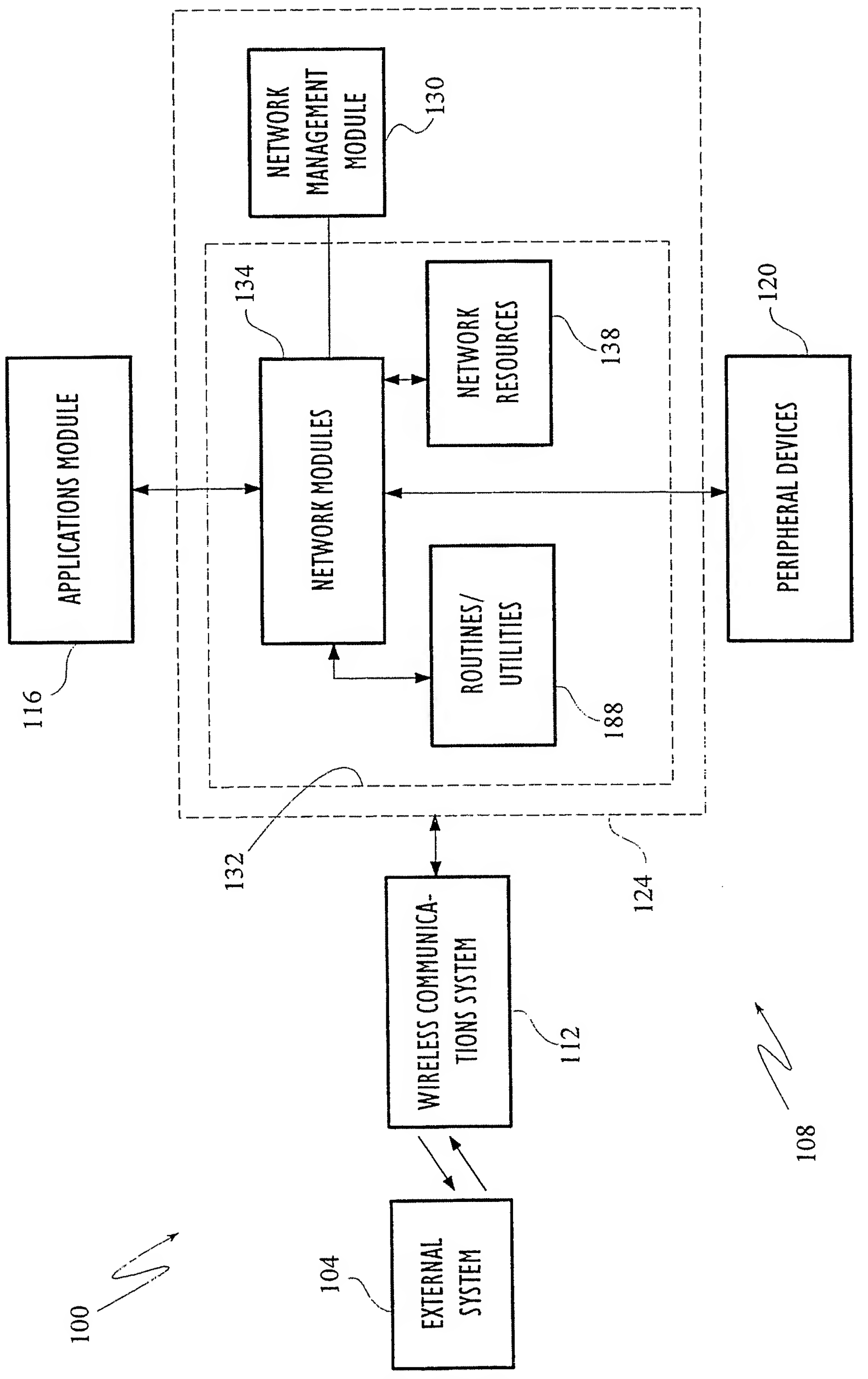
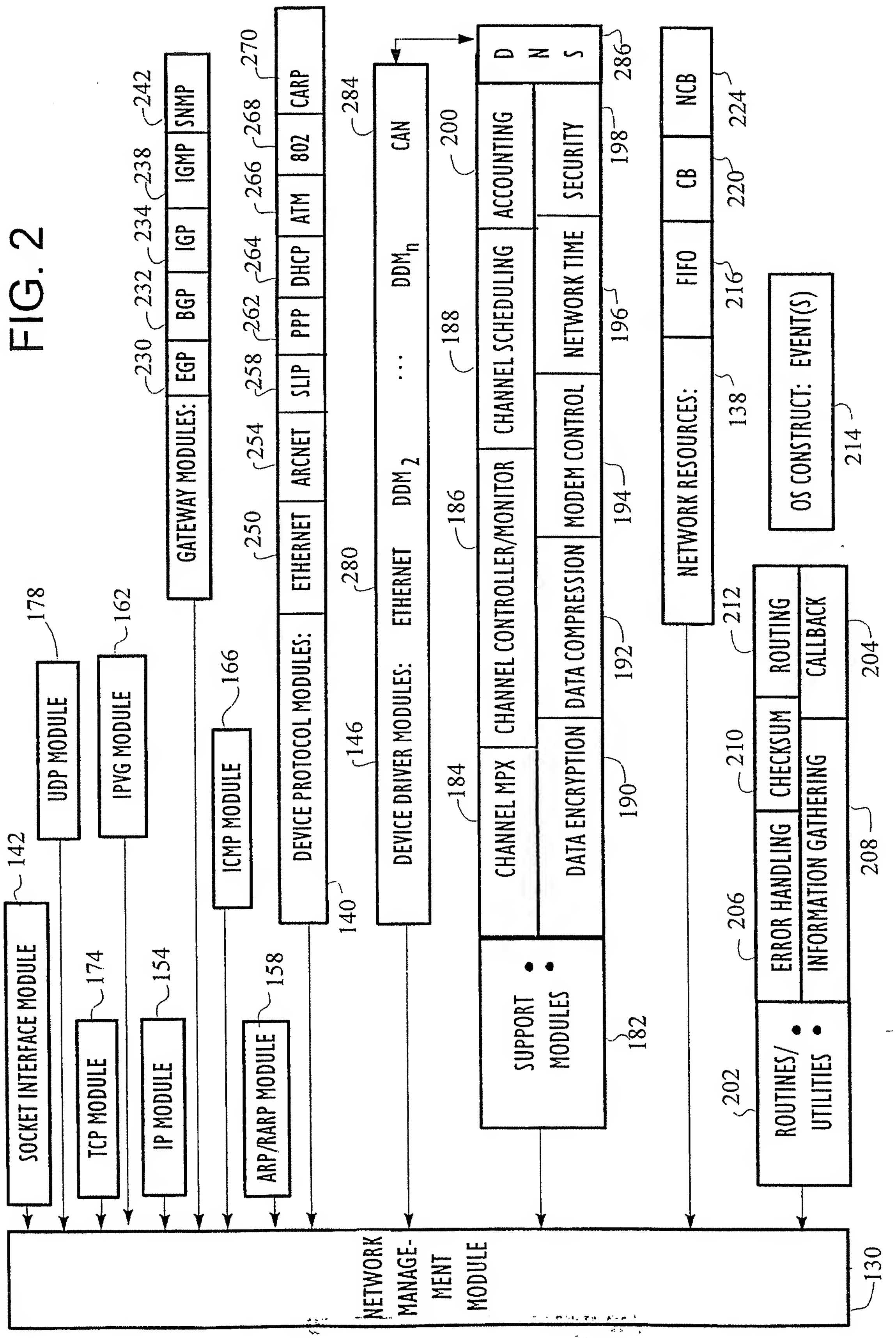
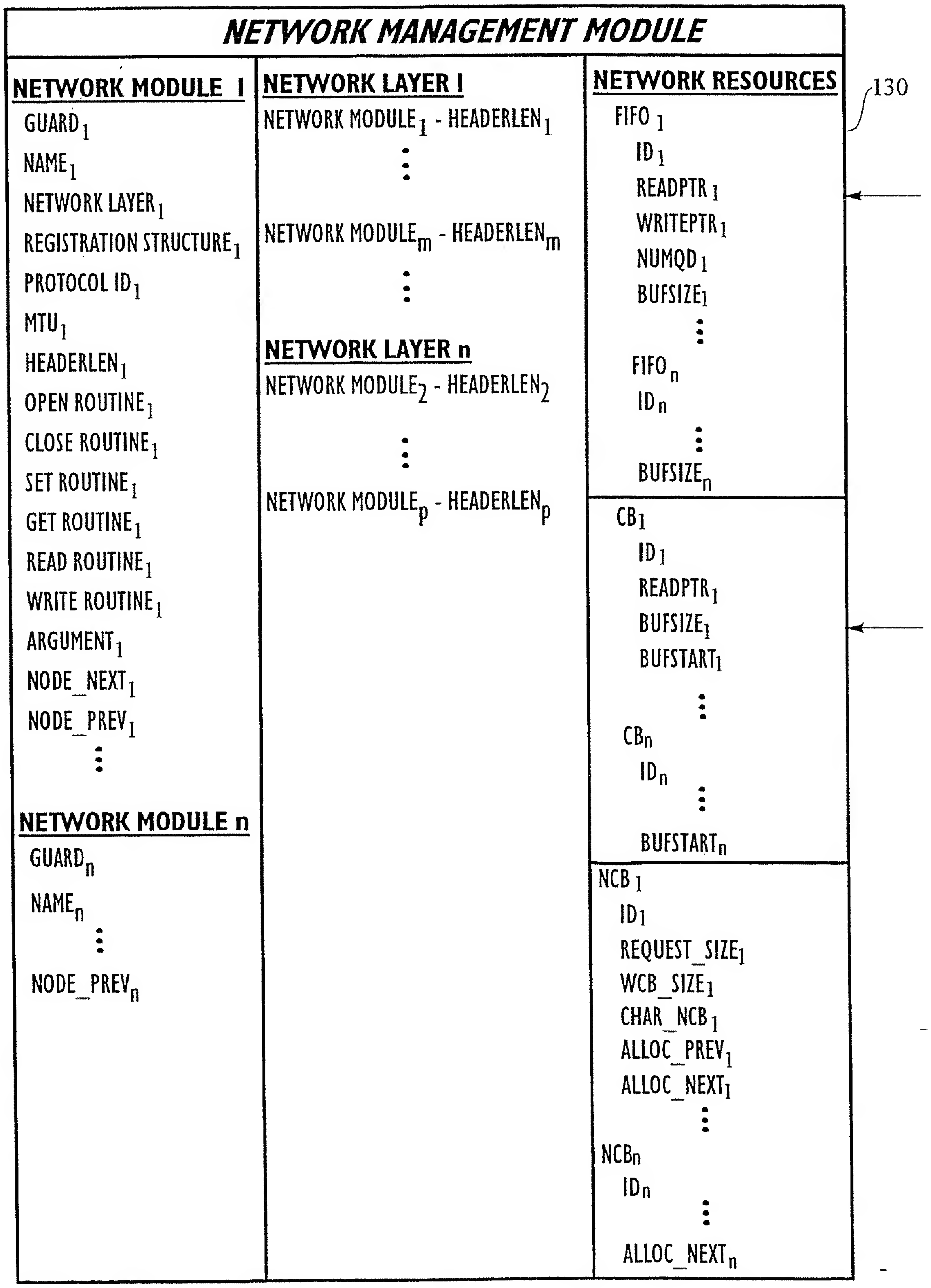

NETWORK MANAGEMENT MODULE
130
134
NETWORK RESOURCES
138
120
APPLICATIONS MODULE
NETWORK MODULES
PERIPHERAL DEVICES
ROUTINES/ UTILITIES
116
188
132
124
WIRELESS COMMUNICA-TIONS SYSTEM
112
108
EXTERNAL SYSTEM
104
100

FIG. 1

FIG. 2


NETWORK MANAGE-MENT MODULE
130
SOCKET INTERFACE MODULE
142
UDP MODULE
178
TCP MODULE
174
IPVG MODULE
162
IP MODULE
154
ICMP MODULE
166
ARP/RARP MODULE
158
GATEWAY MODULES:
EGP
230
BGP
232
IGP
234
IGMP
238
SNMP
242
DEVICE PROTOCOL MODULES:
140
ETHERNET
250
ARCNET
254
SLIP
258
PPP
262
DHCP
264
ATM
266
802
268
CARP
270
DEVICE DRIVER MODULES:
146
ETHERNET
280
DDM 2
...
DDM n
CAN
284
SUPPORT MODULES
182
CHANNEL MPX
184
CHANNEL CONTROLLER/MONITOR
186
CHANNEL SCHEDULING
188
ACCOUNTING
200
DNS
286
DATA ENCRYPTION
190
DATA COMPRESSION
192
MODEM CONTROL
194
NETWORK TIME
196
SECURITY
198
NETWORK RESOURCES:
138
FIFO
216
CB
220
NCB
224
OS CONSTRUCT: EVENT(S)
214
ROUTINES/ UTILITIES
202
ERROR HANDLING
206
CHECKSUM
210
ROUTING
212
INFORMATION GATHERING
208
CALLBACK
204



NETWORK MANAGEMENT MODULE
NETWORK MODULE I
GUARD1
NAME1
NETWORK LAYER1
REGISTRATION STRUCTURE1
PROTOCOL ID1
MTU1
HEADERLEN1
OPEN ROUTINE1
CLOSE ROUTINE1
SET ROUTINE1
GET ROUTINE1
READ ROUTINE1
WRITE ROUTINE1
ARGUMENT1
NODE_NEXT1
NODE_PREV1
NETWORK MODULE n
GUARDn
NAMEn
NODE_PREVn
NETWORK LAYER I
NETWORK MODULE1 - HEADERLEN1
NETWORK MODULEm - HEADERLENm
NETWORK LAYER n
NETWORK MODULE2 - HEADERLEN2
NETWORK MODULEp - HEADERLENp
NETWORK RESOURCES
130
FIFO 1
ID1
READPTR1
WRITEPTR1
NUMQD1
BUFSIZE1
FIFO n
IDn
BUFSIZEn
CB1
ID1
READPTR1
BUFSIZE1
BUFSTART1
CBn
IDn
BUFSTARTn
NCB 1
ID1
REQUEST_SIZE1
WCB_SIZE1
CHAR_NCB1
ALLOC_PREV1
ALLOC_NEXT1
NCBn
IDn
ALLOC_NEXTn

FIG. 3A

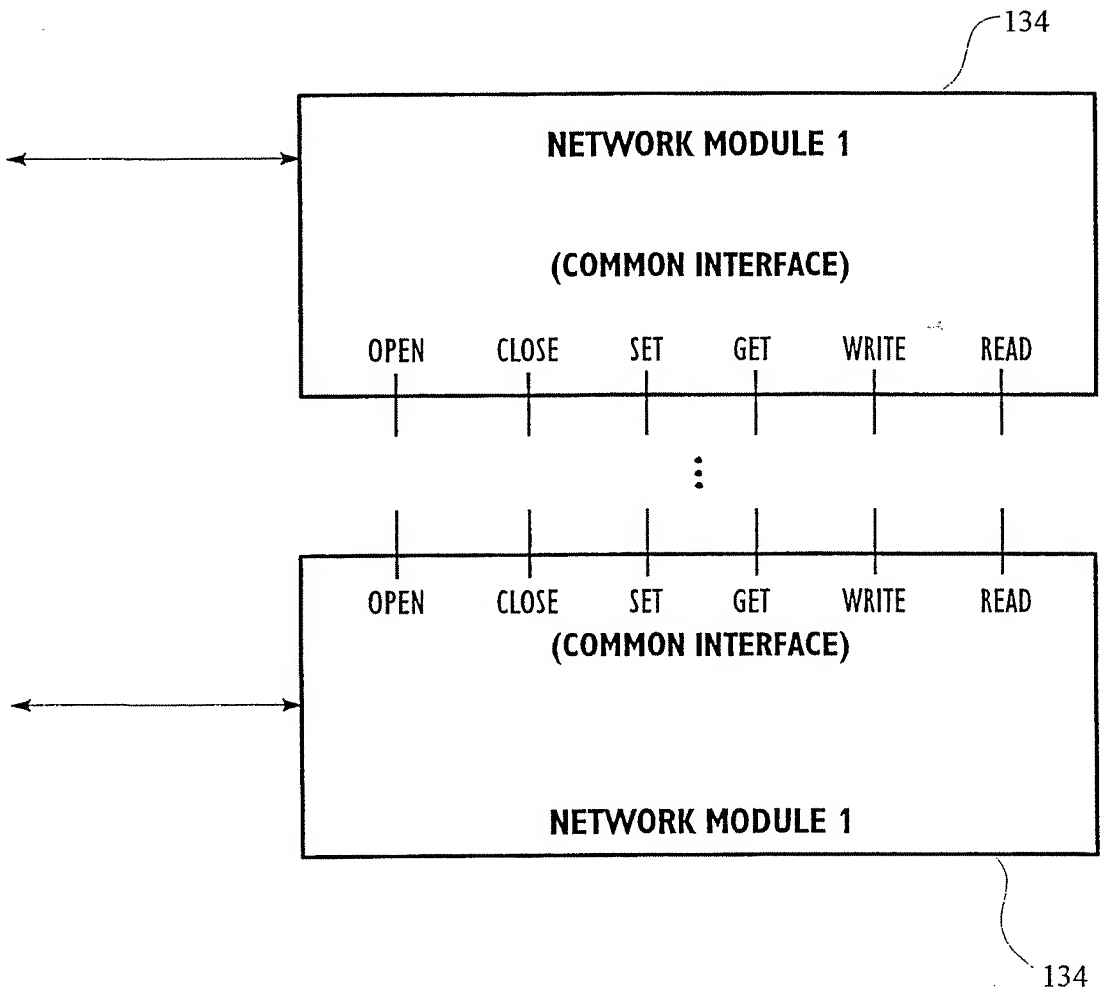

134
NETWORK MODULE 1
(COMMON INTERFACE)
OPEN
CLOSE
SET
GET
WRITE
READ
OPEN
CLOSE
SET
GET
WRITE
READ
(COMMON INTERFACE)
NETWORK MODULE 1
134

FIG. 3B

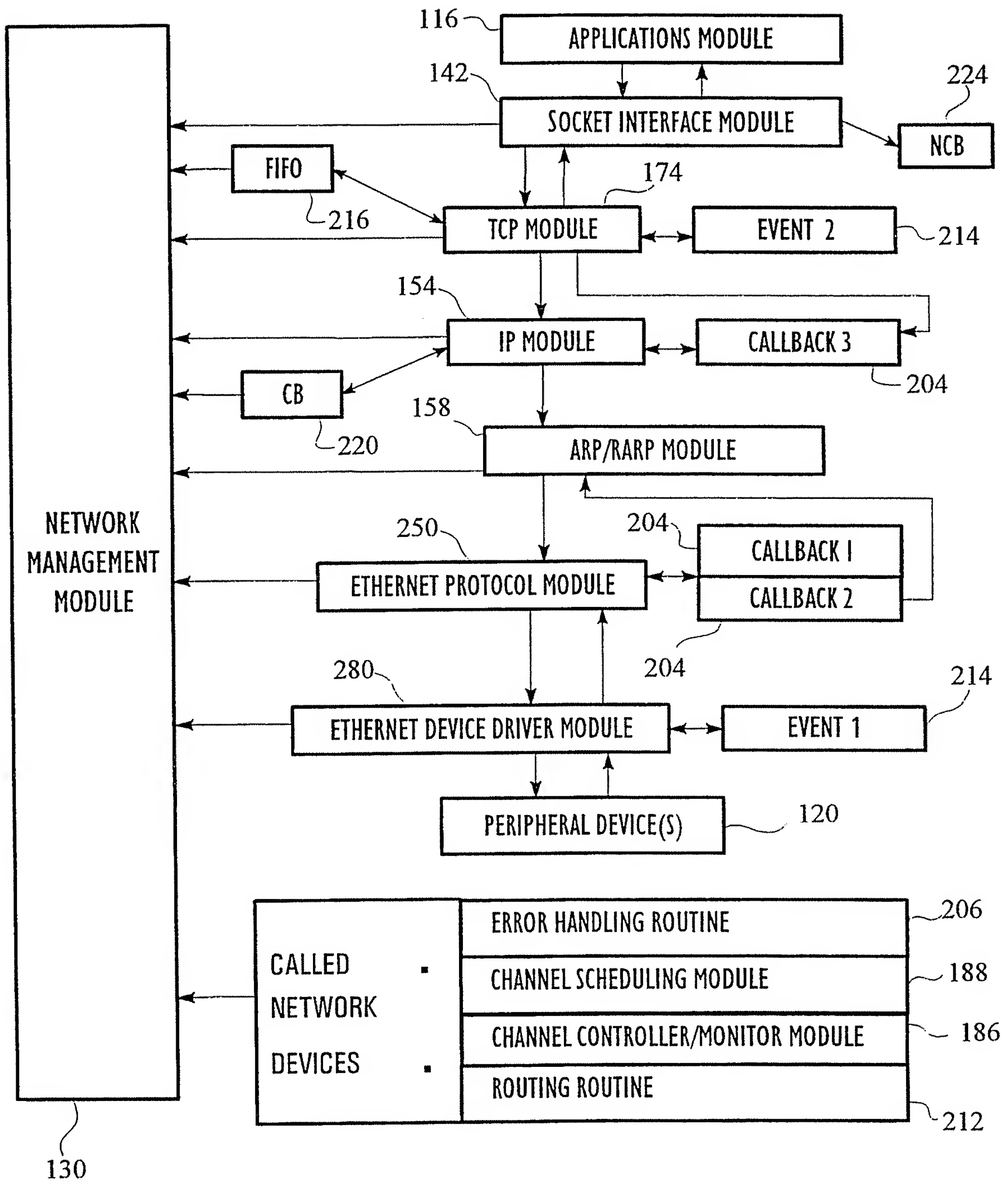

APPLICATIONS MODULE
116
SOCKET INTERFACE MODULE
142
NCB
224
FIFO
216
TCP MODULE
174
EVENT 2
214
154
IP MODULE
CALLBACK 3
204
CB
220
158
ARP/RARP MODULE
NETWORK MANAGEMENT MODULE
250
ETHERNET PROTOCOL MODULE
204
CALLBACK 1
CALLBACK 2
204
280
ETHERNET DEVICE DRIVER MODULE
EVENT 1
214
PERIPHERAL DEVICE(S)
120
CALLED NETWORK DEVICES
ERROR HANDLING ROUTINE
206
CHANNEL SCHEDULING MODULE
188
CHANNEL CONTROLLER/MONITOR MODULE
186
ROUTING ROUTINE
212
130

FIG. 4

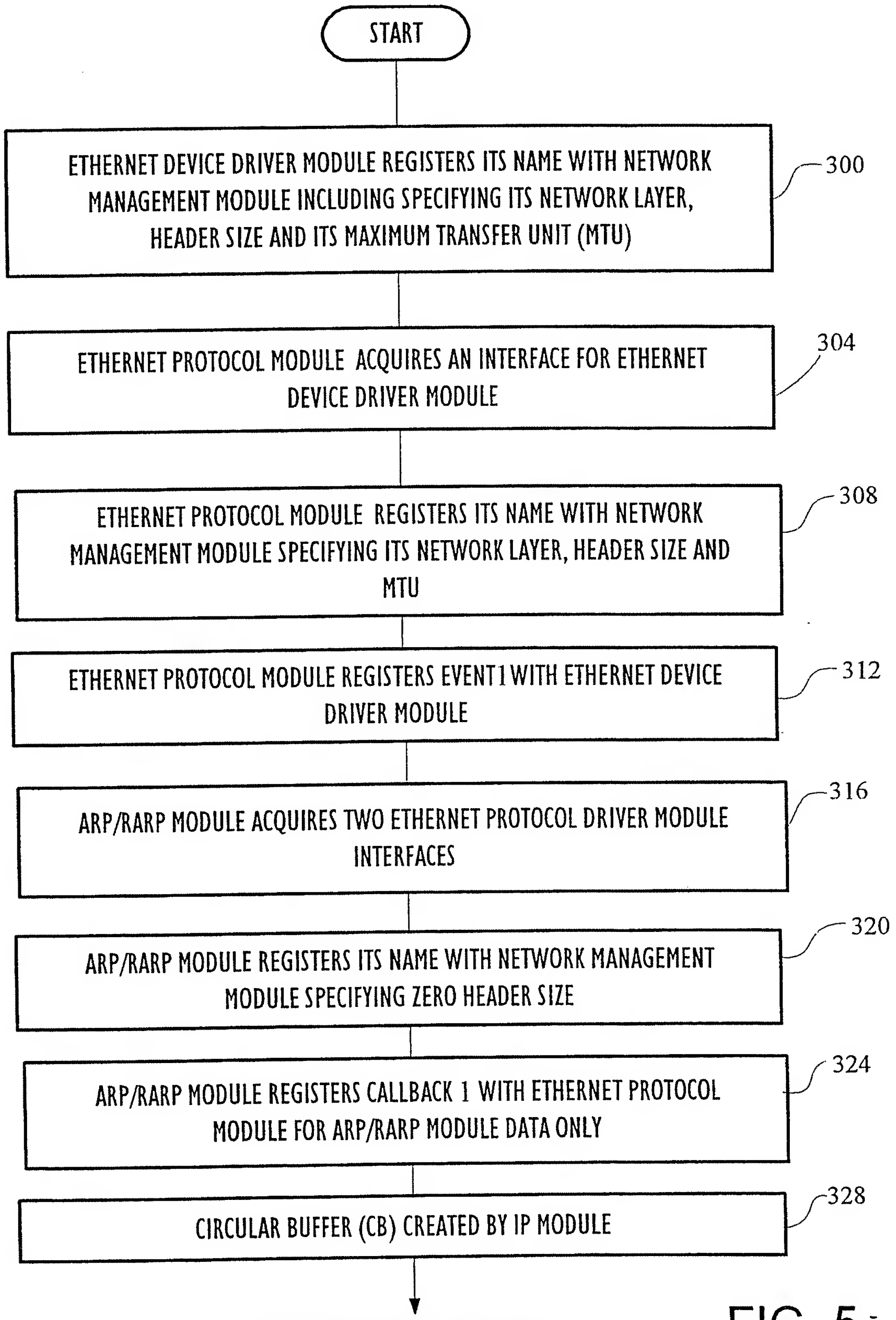

START
ETHERNET DEVICE DRIVER MODULE REGISTERS ITS NAME WITH NETWORK MANAGEMENT MODULE INCLUDING SPECIFYING ITS NETWORK LAYER, HEADER SIZE AND ITS MAXIMUM TRANSFER UNIT (MTU)
300
ETHERNET PROTOCOL MODULE ACQUIRES AN INTERFACE FOR ETHERNET DEVICE DRIVER MODULE
304
ETHERNET PROTOCOL MODULE REGISTERS ITS NAME WITH NETWORK MANAGEMENT MODULE SPECIFYING ITS NETWORK LAYER, HEADER SIZE AND MTU
308
ETHERNET PROTOCOL MODULE REGISTERS EVENT1 WITH ETHERNET DEVICE DRIVER MODULE
312
ARP/RARP MODULE ACQUIRES TWO ETHERNET PROTOCOL DRIVER MODULE INTERFACES
316
ARP/RARP MODULE REGISTERS ITS NAME WITH NETWORK MANAGEMENT MODULE SPECIFYING ZERO HEADER SIZE
320
ARP/RARP MODULE REGISTERS CALLBACK 1 WITH ETHERNET PROTOCOL MODULE FOR ARP/RARP MODULE DATA ONLY
324
CIRCULAR BUFFER (CB) CREATED BY IP MODULE
328

FIG. 5

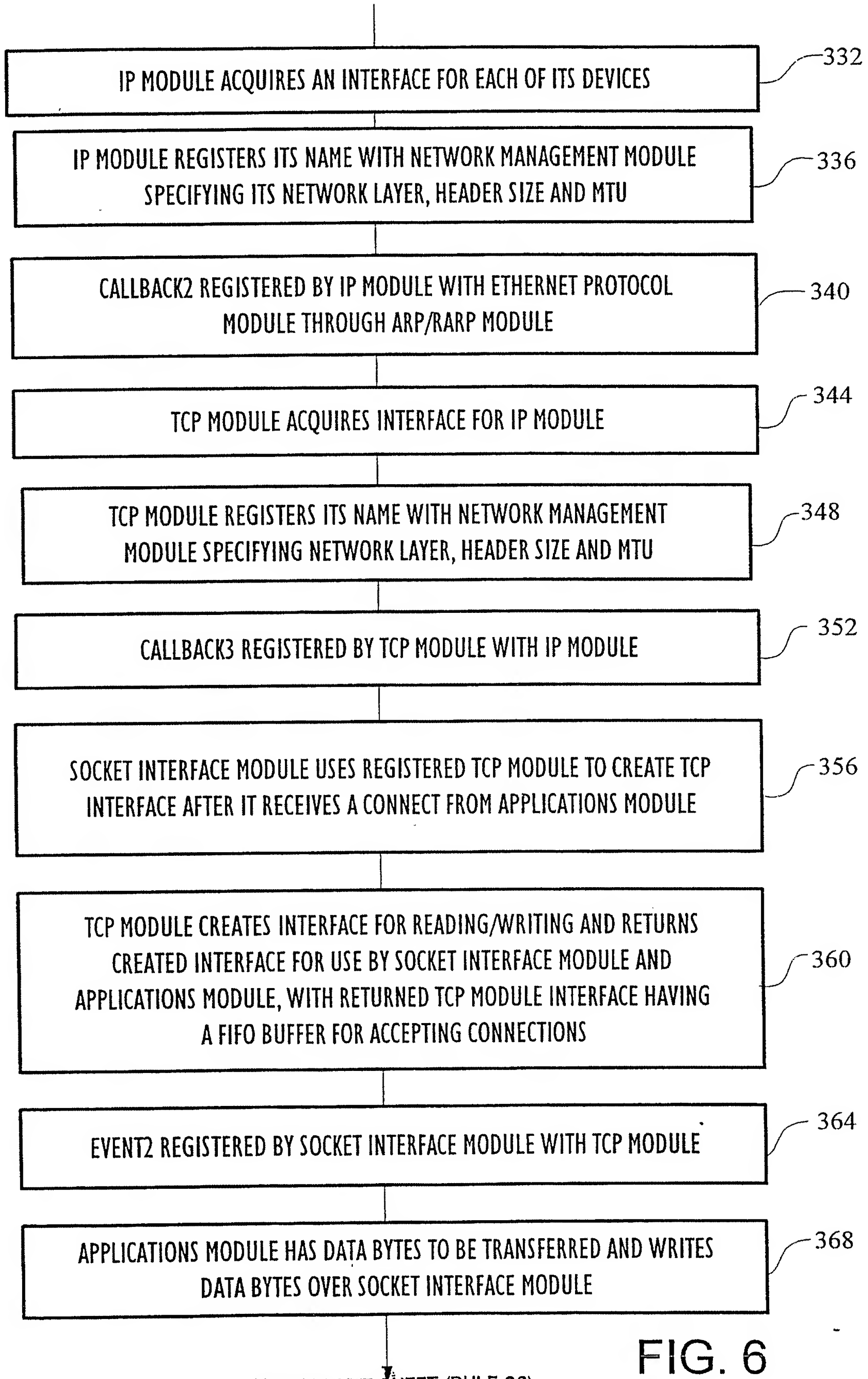

IP MODULE ACQUIRES AN INTERFACE FOR EACH OF ITS DEVICES
332
IP MODULE REGISTERS ITS NAME WITH NETWORK MANAGEMENT MODULE SPECIFYING ITS NETWORK LAYER, HEADER SIZE AND MTU
336
CALLBACK2 REGISTERED BY IP MODULE WITH ETHERNET PROTOCOL MODULE THROUGH ARP/RARP MODULE
340
TCP MODULE ACQUIRES INTERFACE FOR IP MODULE
344
TCP MODULE REGISTERS ITS NAME WITH NETWORK MANAGEMENT MODULE SPECIFYING NETWORK LAYER, HEADER SIZE AND MTU
348
CALLBACK3 REGISTERED BY TCP MODULE WITH IP MODULE
352
SOCKET INTERFACE MODULE USES REGISTERED TCP MODULE TO CREATE TCP INTERFACE AFTER IT RECEIVES A CONNECT FROM APPLICATIONS MODULE
356
TCP MODULE CREATES INTERFACE FOR READING/WRITING AND RETURNS CREATED INTERFACE FOR USE BY SOCKET INTERFACE MODULE AND APPLICATIONS MODULE, WITH RETURNED TCP MODULE INTERFACE HAVING A FIFO BUFFER FOR ACCEPTING CONNECTIONS
360
EVENT2 REGISTERED BY SOCKET INTERFACE MODULE WITH TCP MODULE
364
APPLICATIONS MODULE HAS DATA BYTES TO BE TRANSFERRED AND WRITES DATA BYTES OVER SOCKET INTERFACE MODULE
368

FIG. 6

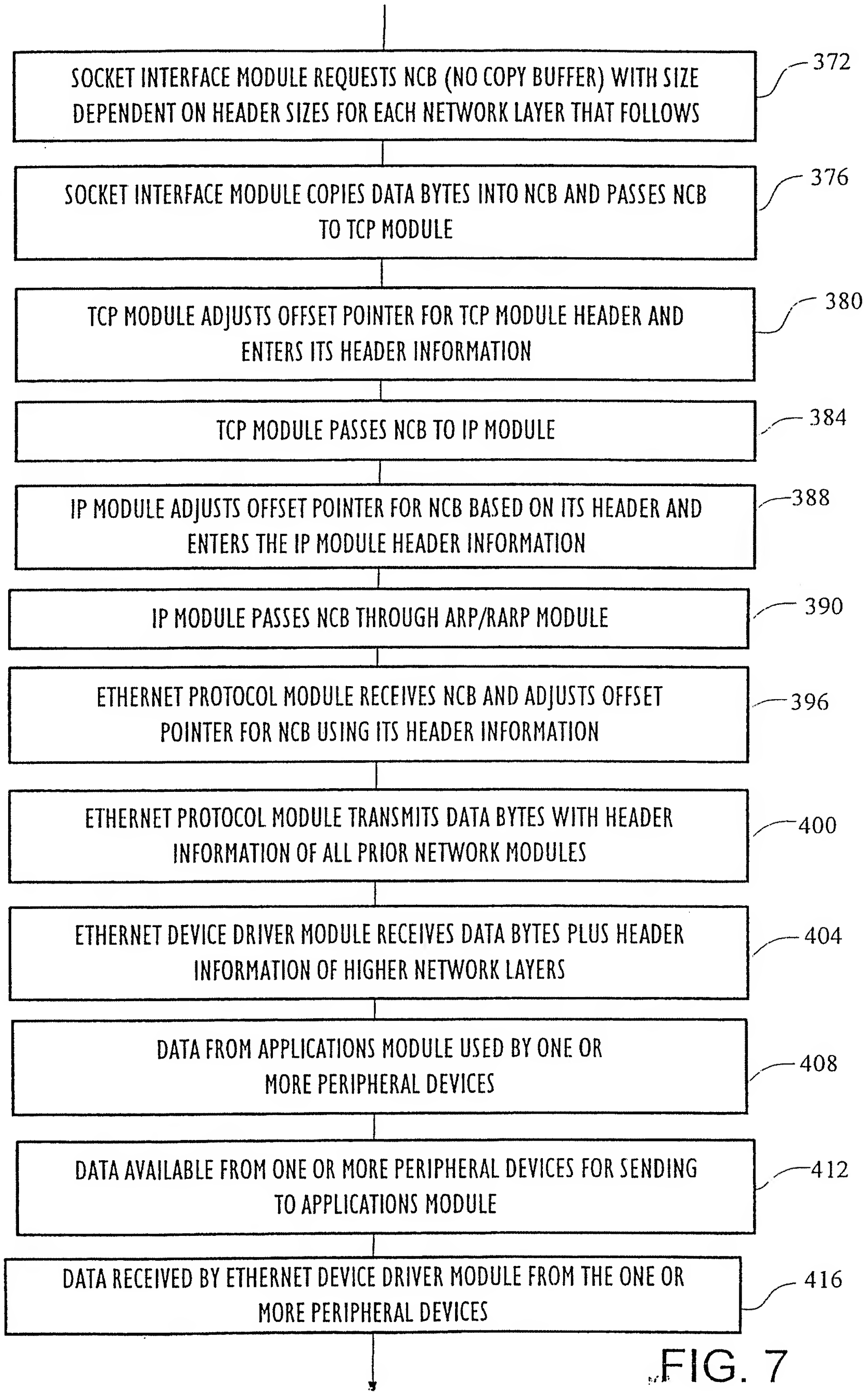

SOCKET INTERFACE MODULE REQUESTS NCB (NO COPY BUFFER) WITH SIZE DEPENDENT ON HEADER SIZES FOR EACH NETWORK LAYER THAT FOLLOWS
372
SOCKET INTERFACE MODULE COPIES DATA BYTES INTO NCB AND PASSES NCB TO TCP MODULE
376
TCP MODULE ADJUSTS OFFSET POINTER FOR TCP MODULE HEADER AND ENTERS ITS HEADER INFORMATION
380
TCP MODULE PASSES NCB TO IP MODULE
384
IP MODULE ADJUSTS OFFSET POINTER FOR NCB BASED ON ITS HEADER AND ENTERS THE IP MODULE HEADER INFORMATION
388
IP MODULE PASSES NCB THROUGH ARP/RARP MODULE
390
ETHERNET PROTOCOL MODULE RECEIVES NCB AND ADJUSTS OFFSET POINTER FOR NCB USING ITS HEADER INFORMATION
396
ETHERNET PROTOCOL MODULE TRANSMITS DATA BYTES WITH HEADER INFORMATION OF ALL PRIOR NETWORK MODULES
400
ETHERNET DEVICE DRIVER MODULE RECEIVES DATA BYTES PLUS HEADER INFORMATION OF HIGHER NETWORK LAYERS
404
DATA FROM APPLICATIONS MODULE USED BY ONE OR MORE PERIPHERAL DEVICES
408
DATA AVAILABLE FROM ONE OR MORE PERIPHERAL DEVICES FOR SENDING TO APPLICATIONS MODULE
412
DATA RECEIVED BY ETHERNET DEVICE DRIVER MODULE FROM THE ONE OR MORE PERIPHERAL DEVICES
416

FIG. 7

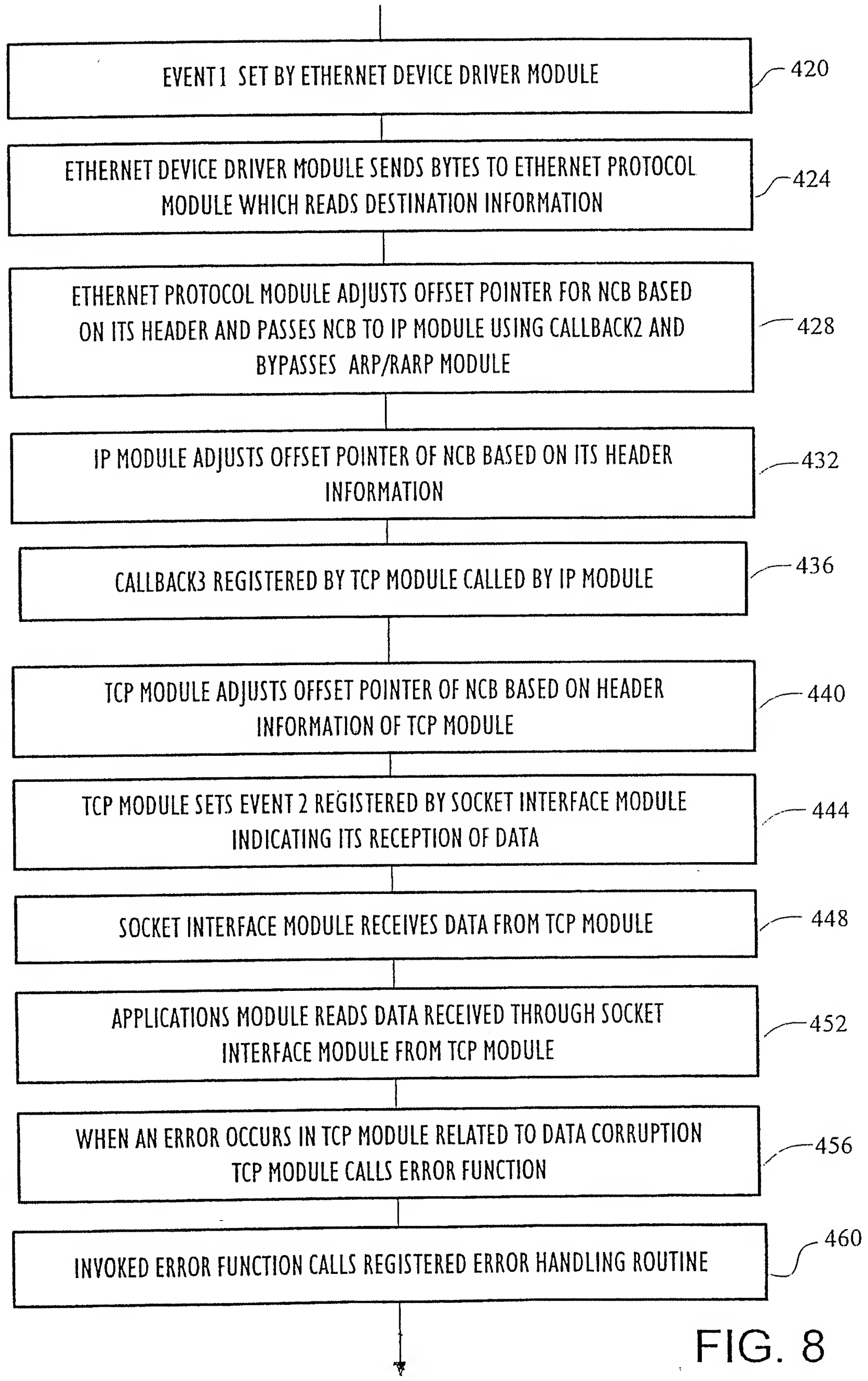

EVENT1 SET BY ETHERNET DEVICE DRIVER MODULE
420
ETHERNET DEVICE DRIVER MODULE SENDS BYTES TO ETHERNET PROTOCOL MODULE WHICH READS DESTINATION INFORMATION
424
ETHERNET PROTOCOL MODULE ADJUSTS OFFSET POINTER FOR NCB BASED ON ITS HEADER AND PASSES NCB TO IP MODULE USING CALLBACK2 AND BYPASSES ARP/RARP MODULE
428
IP MODULE ADJUSTS OFFSET POINTER OF NCB BASED ON ITS HEADER INFORMATION
432
CALLBACK3 REGISTERED BY TCP MODULE CALLED BY IP MODULE
436
TCP MODULE ADJUSTS OFFSET POINTER OF NCB BASED ON HEADER INFORMATION OF TCP MODULE
440
TCP MODULE SETS EVENT 2 REGISTERED BY SOCKET INTERFACE MODULE INDICATING ITS RECEPTION OF DATA
444
SOCKET INTERFACE MODULE RECEIVES DATA FROM TCP MODULE
448
APPLICATIONS MODULE READS DATA RECEIVED THROUGH SOCKET INTERFACE MODULE FROM TCP MODULE
452
WHEN AN ERROR OCCURS IN TCP MODULE RELATED TO DATA CORRUPTION TCP MODULE CALLS ERROR FUNCTION
456
INVOKED ERROR FUNCTION CALLS REGISTERED ERROR HANDLING ROUTINE
460

FIG. 8

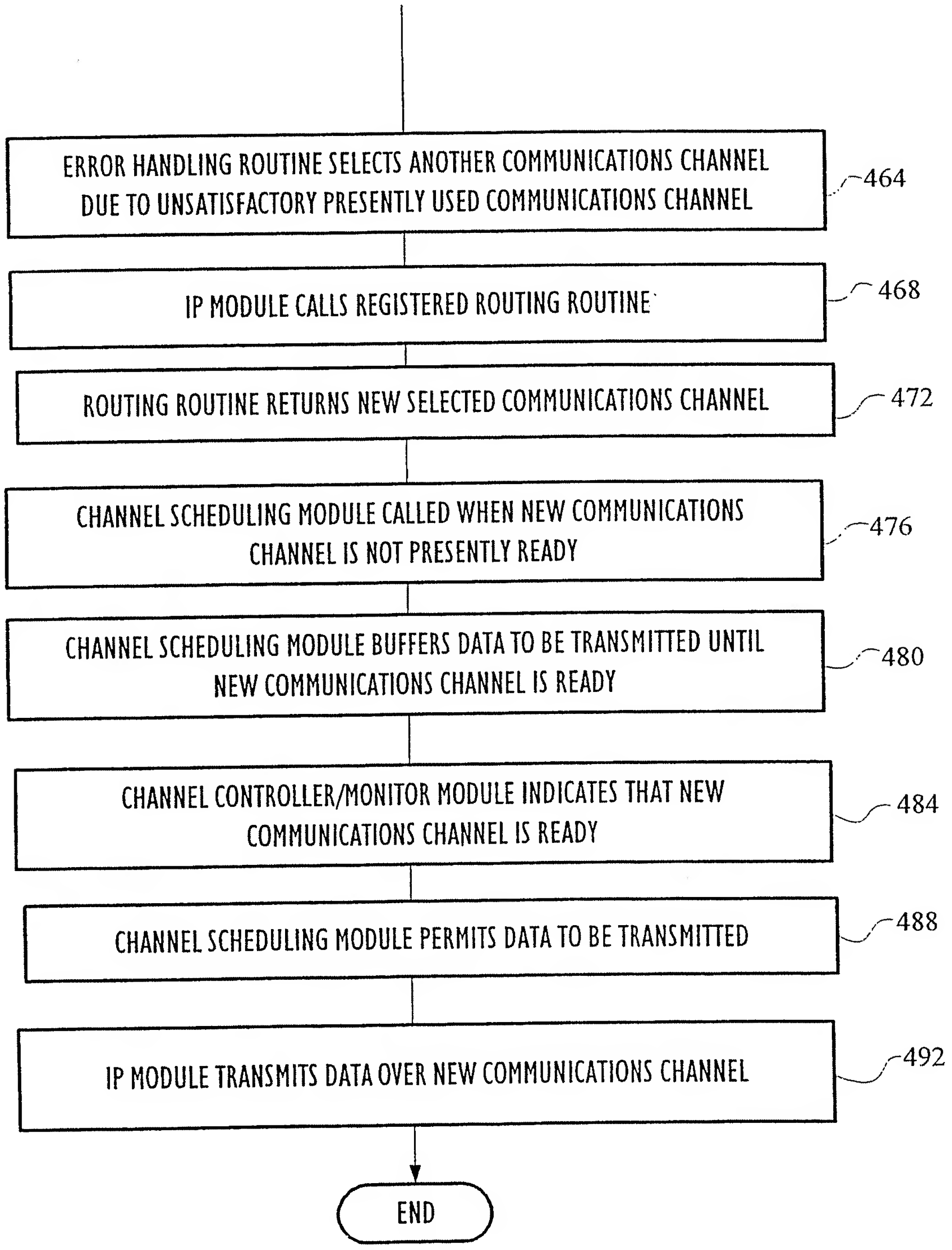

ERROR HANDLING ROUTINE SELECTS ANOTHER COMMUNICATIONS CHANNEL DUE TO UNSATISFACTORY PRESENTLY USED COMMUNICATIONS CHANNEL
464
IP MODULE CALLS REGISTERED ROUTING ROUTINE
468
ROUTING ROUTINE RETURNS NEW SELECTED COMMUNICATIONS CHANNEL
472
CHANNEL SCHEDULING MODULE CALLED WHEN NEW COMMUNICATIONS CHANNEL IS NOT PRESENTLY READY
476
CHANNEL SCHEDULING MODULE BUFFERS DATA TO BE TRANSMITTED UNTIL NEW COMMUNICATIONS CHANNEL IS READY
480
CHANNEL CONTROLLER/MONITOR MODULE INDICATES THAT NEW COMMUNICATIONS CHANNEL IS READY
484
CHANNEL SCHEDULING MODULE PERMITS DATA TO BE TRANSMITTED
488
IP MODULE TRANSMITS DATA OVER NEW COMMUNICATIONS CHANNEL
492
END

FIG. 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/22598

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 13/00
US CL : 709/200, 301
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 709/200, 223, 227, 230, 231, 301, 302

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Microsoft Press Computer Dictionary, TCP/IP Illustrated, Volumes 1-2

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS, WEST
search terms: network management module, device driver, OSI, network layer, registration

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,636,371 A (YU) 03 June 1997, abstract, col. 9, line 23 - col. 12, line 54	1-34
Y	US 5,630,061 A (RICHTER et al.) 13 May 1997, abstract, col. 5, line 29 - col. 12, line 67	1-34
A	WRIGHT, Gary R. et al., TCP/IP Illustrated Volume 2: The Implementation, Addison-Wesley Professional Computing Series, March 1996, pages 63-74, especially 65-66.	1-34

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A" document defining the general state of the art which is not considered to be of particular relevance	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E" earlier document published on or after the international filing date	
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	"&" document member of the same patent family

Date of the actual completion of the international search 26 FEBRUARY 1999	Date of mailing of the international search report 12 MAR 1999
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer ZARNI MAUNG [illegible] Telephone No. (703) 308-6687

Form PCT/ISA/210 (second sheet)(July 1992) ★